Thunderspy

Description: This week we examine Firefox's recent move to 76 and slightly beyond; a wonderful new feature coming to Edge; and the security responsibility that attends the use of WordPress, vBulletin, and other complex and sophisticated web applications. We look at the plans for this summer's much-anticipated Black Hat and DEF CON conferences, a newly revealed CRITICAL bug affecting all of the past six years of Samsung Smartphones, and Zoom's latest security-boosting acquisition. I'll then provide an update on my SpinRite work which includes a bit of a rearrangement in sequence to provide another shorter term deliverable. And then we look at the new Thunderspy vulnerability that has the tech press huffing and puffing.

High quality (64 kbps) mp3 audio file URL: http://media.GRC.com/sn/SN-766.mp3
Quarter size (16 kbps) mp3 audio file URL: http://media.GRC.com/sn/sn-766-lq.mp3

SHOW TEASE: It's time for Security Now!. Steve Gibson is here. Coming up, Firefox adds a new password manager, WordPress once again in the crosshairs, Zoom acquires Keybase, and a look at Thunderspy, a new Thunderbolt vulnerability that everyone's subject to. It's all coming up next on Security Now!.

Leo Laporte: This is Security Now! with Steve Gibson, Episode 766, recorded Tuesday, May 12th, 2020: Thunderspy.

It's time for Security Now!, the show where we protect you and your loved ones, your privacy, your security. We even teach you a little bit about technology as we talk. That's because of this guy here, the best teacher ever. He's like the teacher you had in 10th grade you'll never forget, taught you everything you ever wanted to know about electronics: Mr. Steve Gibson. Hi, Steve.

Steve Gibson: Yeah, mine, his name was Harold Fearon. He was my high school electronics teacher. And he did make my...

Leo: You don't ever forget them, do you.

Steve: No. Quite an impression. Everybody hated him except I liked him. He was ruthless and rigorous and...

Leo: Perfect.

Steve: But if you played by the rules, it was all good.

Leo: It's like Mr. Devore, my chemistry teacher. I loved him. Got an A in chemistry because of him. You're right. You never forget those guys.

Steve: Yeah.

Leo: So, hello, Steve.

Steve: So we're going to talk about Thunderspy, which is the name given to the result of an analysis by a security researcher into the dilemma essentially that we currently have, which is that PC designers seem incapable of not routing the system's internal bus to the outside. Which is just a bad idea. But that's what we have. Anyway, lots to talk about. We're going to examine Firefox's recent move to its release 76 and slightly beyond; a wonderful new feature coming to Edge; the security responsibility that attends the use of WordPress, vBulletin, and other complex and sophisticated web applications, with examples from this past week.

We look at the plans for this summer's much anticipated Black Hat and DEF CON conferences, a newly revealed critical bug affecting all of the past six years of Samsung's smartphones, and the need to patch if you think you might be a target of opportunity. And unfortunately - I know you're already mourning this, Leo - Zoom's latest security-boosting acquisition.

Leo: Yeah. Bad for me, but probably very good for Zoom, I'm guessing.

Steve: Yes, it was brilliant because from a PR standpoint you couldn't get a better - right now they need headlines more than they need actual technology, so they bought themselves some headlines. I'll then provide an update on my SpinRite work.

Leo: Oh, good.

Steve: Which includes, as a result of what I've learned since I've been back to it, a bit of a rearrangement in sequence to provide an additional short-term deliverable. And then we're going to look at this new Thunderspy vulnerability suite that has the tech press all huffing and puffing. So I think another great podcast. Oh, and Leo, we've got a Picture of the Week for the ages.

Leo: I just saw it. I'm laughing. I am laughing and laughing and laughing. Good. So, Steve Gibson.

Steve: So our Picture of the Week...

Leo: Oh, my god, that's hysterical.

Steve: It was tweeted to me. I checked into Twitter as I often do when I'm looking for something, to see if someone has said, "Hey, Steve, how's this as a candidate?" And I love it because it had to have been a setup, I presume. But still, it's no less funny. Of course we're all familiar with the old-school practice of hiding a key under the mat. I mean, that's like a meme of the world, hiding a key under the mat.

Leo: I know, and I do it. I wonder if I'm giving away a secret to say that, but that's where we put the key.

Steve: Well, and if it's not there, then you of course look under the pot because then, if there's a pot on the porch, it's going to be underneath the pot. It makes the pot rock a little bit, but that's a giveaway. Anyway, this particular mat has a problem which is it's more non-mat than - or maybe about 50-50 - non-mat than mat. It's a grid of holes connected with some rubber. And the key is showing right through the holes. And for that matter I guess the person's shadow, come to think of it, is also quite apparent as the photo is being taken. Anyway, just as a...

Leo: That's hysterical.

Steve: ...little item of Security Now! interest, I think that's pretty good.

Leo: "The key's under the mat." "I know."

Steve: Uh-huh. Don't even have to lift it to find it.

Leo: No.

Steve: So last week Firefox 76 was released with some new features. This was on last Tuesday while we were doing the podcast. And this is for the desktop OSes - Windows, Mac, and Linux. They fixed a bunch of bugs we'll get to in a second, and added a couple nice new features. It was around this time last year that Mozilla first began sort of sticking their toe in the water with their own password manager extension called Lockwise. It's homegrown. And today, a year later, I mean, we've mentioned it a couple times in passing. We all know I don't use it. But there is one from Mozilla.

Of course it's probably Firefox, I'm sure it's Firefox-only because it's implemented as a Firefox extension. So you don't get the multiplatform, multibrowser benefit of a solution which is cross-browser. But if you are a Firefox-only person, maybe it works for you. So presumably it's acquired a following, and it has been continuing to get support and some future growth.

So they've just added a couple nice features, clearly tying it into, probably, I didn't see this explicitly, but I would imagine it's tied into Troy Hunt's Have I Been Pwned database which, as we know, he's made an API available for this kind of purpose. You will now, if you are a Firefox Lockwise user, you get a notification of a website breach on the Internet that may be affecting you. A notice is presented saying that passwords were leaked or stolen from this website since you last updated your login details. Change your password to protect your account. So a handy feature.

And in a little bit of a variation on that, they have a different approach, a different notice that you can get saying this password has been used on another account that was likely in a data breach. Again, the password sharing problem. And they say reusing credentials puts all your accounts at risk. Change this password. So anyway, some nice feature additions. No biggie, but worthwhile.

And in a nice security-tightening measure, before displaying a user's saved login credentials, Lockwise will now fall back to requiring any users who have not previously established a master password for Lockwise to reenter their local computer's login account password. So that seems like a good thing. This will prevent anyone who might obtain access to your logged-in computer session from snooping into your stored username and passwords. I have freaked out several users of Chrome by showing them - and they're all, yeah, well, this is really a secure browser. I show them how easy it is to have Chrome unmask all of the passwords they have asked their Chrome browser to save for them. Virtually no one appreciates that the entire list of saved usernames and passwords is displayable after a few clicks.

Leo: And that's true in Chrome, too. That's why we've always said don't use a browser to keep your passwords.

Steve: Yes, yes, yes, exactly. We also previously touched on a planned feature in Firefox which dropped in this version. We said that they would be adding a video pop-out feature which allows you to lift a page's playing video off away from the page into a separate floating window. And that feature went live last week with Firefox 76. BleepingComputer noted in their coverage of this that the feature didn't yet appear to be working reliably. I think rather than just talking about it, they actually tried to use it, and some pages would only allow one to pop out, then nothing else could be popped out. Or like it only happened once per browser session. I mean, it was like not ready yet. So I imagine that it'll be getting some more, becoming more solid in the future with additional iterations.

I mentioned some security vulnerabilities. They fixed 11 CVE-numbered vulnerabilities, three of which were rated critical. There was a use-after-free during worker shutdown. And as we all know, any allowance of the use of freed memory is never good. So Mozilla notes that this results in a "potentially exploitable" crash. They don't know that it could be. It's not in the wild so it's not a zero-day. But good to have that fixed.

There's also a sandbox escape with improperly guarded access tokens. And of course since a sandbox escape means that a web page's rogue content, if there were any, might be able to escape its confinement and interfere either with other pages or with the host operating system, that's not good either. So that's gone. And then there's a sort of a catchall. They just said "memory safety bugs," plural, fixed in Firefox 76 and their extended service release 68.8. So they're not specifying what they were, but we know that there were eight different subtle memory bugs that were found and smashed. So good for that.

On the other hand, shortly after it first appeared, Mozilla encountered, or I should say their users encountered, a pair of newly introduced bugs that it could not ignore, Mozilla could not ignore. One caused the Amazon Assistant browser extension to no longer function correctly, and the other fixed an obscure browser crash in Windows 7 32-bit systems with Nvidia graphics. Go figure. Anyway, they paused 76's ongoing rollout as soon as those things appeared because they thought, okay, well, we can't keep doing this. They fixed those.

And then last Friday, three days after the release, 76.0.1 became available now to all Firefox users. And since I run with an instance of Firefox open permanently and

maximized in my lower left-hand screen, none of those updates were happening here. So when I read about this in preparation for the podcast, I went over to About Firefox and was immediately asked for permission to restart Firefox, which I did, of course. And now I have 76.0.1. So these are not critical. They're not zero-day. As far as we know they're not exploited in the wild. I didn't dig into how they learned of them. But if you are like me, and your Firefox just sits there happily open all the time, and also like me you never typically reboot your system, I mean, mine goes for, well, almost years at a time since it's pretty stable. You just may want to go over there and have Firefox fix itself because that's good.

So I was very excited about this when I first read about it because I thought it was unique. [Buzzer sound]. But still, it's good to have it. Edge is adding a feature to silence those annoying notification requests. When I'm in research mode, as I have been recently when I'm doing fact-finding on the 'Net for ideas about the best ways to accomplish some programming tasks that I haven't yet encountered before, I tend to roam around, maybe straying more than I usually do from my very boring beaten path.

So the other day I encountered a site that - I love this. I just shook my head. It explained that in order to enable whatever it was that I wanted to do, and I don't really remember what that was, probably download something or maybe click a link and look at some information, I needed to "enable downloads" or "enable compatibility" or some nonsense by clicking the Allow button on the notifications dropdown. And sure enough, there it was, that dropdown that we're seeing now with increasing regularity, hanging down below the URL, asking for me to either allow or block any and all of that website's future push notifications.

We've talked about this annoyance before. It is unfortunately new, relatively new, and extremely abuse prone. It allows websites to which you have given permission to, from then on, interact with your operating system, linking into your OS's native notification system to bother you anytime the site wishes to. Now, just tell me that is not subject to abuse. You know, some ideas, this and like now always needing to get our permission to use cookies, are just very bad. This whole background website notification system is another bad idea.

So blessedly, Edge is getting a feature, and what I wrote here in the show notes is that I sure hope the two browsers I actually use, Firefox and Chrome, will immediately adopt. Under Site Permissions/Notifications on Edge is "Quiet Notification Requests," and it describes itself as "This will prevent notification requests from interrupting you." And with Quiet Notification Requests enabled, this whole annoying permission dropdown is replaced in Edge by a small and easy-to-ignore bell icon which appears at the far right-hand side of the URL address bar.

And if for some reason you should want to see the notification, for example, if you're being held ransom over accepting notifications before the site will do what you want, you can click the bell to manage notifications - that's one of the options - or allow them for the site. And Edge also provides a very clean Manage facility that would allow you to easily revoke any notification permissions you may have come to regret.

The moral of this story is that sometimes you need to go looking for what you want. After writing this, I became curious about Chrome and Firefox. And I'm glad I did because both of those browsers already offer the functional equivalent of this feature. You've just got to go find it and turn it on. It's now enabled on both of the browsers I use daily. Open either browser's Settings and use the page's search function to search for notifications, and you'll be taken to that function in the UI. Works great.

Leo: First thing I do on both Chrome and Firefox is turn off - and this blocks, it's actually better, or maybe worse, depending on what you want from notifications. I just turn it on. I never see any requests for notifications, so I never have to worry about it one way or the other. And it will give you a list of ones you've already said yes to in case you want to just delete all of those, as well.

Steve: Yeah, yeah. Both browsers have good management, and they both completely mute any requests. Now, the problem we're going to run into, those of us who have done this, is that there are an increasing number of sites, I hit one myself, I was reading about one in this coverage, that in typical slimy site mode where taking advantage of users who don't know better, the website code is able to detect whether you have notifications enabled for it or not. And so the new annoying site behavior is to require you to enable notifications in order to proceed.

Leo: Bye-bye. I'm never going back to that site.

Steve: Exactly.

Leo: I mean, you'd have to really want that site.

Steve: You've got to be desperate for whatever it is, yes.

Leo: As soon as a site did that I'd say, yeah, I don't think so, sorry, yeah.

Steve: Exactly.

Leo: Wow. That's really screwy that a site would do that. Oh, you have to have notifications.

Steve: Does it surprise?

Leo: No, it doesn't surprise me at all.

Steve: No, and it doesn't say that. It says, you know, you must allow full compatibility or some BS.

Leo: Yeah, yeah, yeah, some line.

Steve: And so the user goes, oh, and looks up there, and hits Allow because...

Leo: Just break the web more. Go ahead.

Steve: It's the equivalent of you must download, you must update your Flash Player, that sort of thing. We're never going to get away from this.

Leo: At least the Edge thing is nice because you could turn it on, but it would quiet it when you wanted to; right? You'd have a little more flexibility than just say I don't want notifications ever.

Steve: Correct. Correct. So yeah, I would turn on "Don't bother me." If you come to a site - because it's going to say it in the middle of the page, you must turn on notifications. Look up there and click Allow. Well, those of us who have muted all the notifications, there's a way, at least in Edge, of selectively saying "Show me the notification." So you could then turn it on, get past this block, and then just go back into the browser's management.

Leo: No notifications, yeah.

Steve: And remove it. So, yeah.

Leo: That's good.

Steve: The fight goes on. Now, if we could just get, Leo, a uniform API for all those incessant "This site uses cookies, okay?" permission banners. Mine would be set to "Curmudgeon," which is short for "Yes, yes, yes, cookies, I get it, now don't ever ask me again."

Leo: Well, and this is an example of overregulation, or an unintended consequence of regulation because this comes from GDPR and European regulation. You've got to notify them if you're going to keep cookies. And now every site in the world, and I wonder how many millions of human hours are wasted clicking okay, okay, okay, okay every time.

Steve: It's so wrong.

Leo: Okay. Yeah, there's nothing we can do about that, yup.

Steve: Okay. We've got WordPress in the crosshairs. WordPress, of course, needs no introduction. It's in use by more than 60 million websites, including more than a third of the top 10 million websites as of this time last year is the latest statistics I could see. As we know, it's a complex system written in PHP which supports third-party add-ons.

Leo: As we know, those three things together mean disaster.

Steve: Exactly. Exactly. The terms "complex" and "third-party plugins"...

Leo: And PHP.

Steve: ...are both fundamentally antithetical to security.

Leo: Yeah.

Steve: So starting near the end of last month, and increasing since, the company Defiant, which are the publishers of the Wordfence security plug-in, from their privileged position in a whole bunch of WordPress sites, they detected somebody running a botnet or a proxy system who began attacking WordPress sites from at least 24,000 separate IP addresses. So it's got to be proxies or a net of bots or something because the IPs were coming in from all over the place. This represented a thirtyfold increase in attack traffic.

For example, just on one day, which was this Sunday before last, May 3rd, Defiant logged more than 20 million attacks against more than half a million websites. They said that the attackers focused mostly on exploiting cross-site scripting (XSS) vulnerabilities and plugins that had already received fixes months or even years ago and had previously been targeted in other attacks. Which suggests, if this was a worthwhile attack, that these sites are just not being fixed. There were five that were most targeted, which they identified, again because they were in a position. They're like a WordPress firewall.

There was a cross-site scripting vulnerability in the Easy2Map plugin, which was removed from the WordPress plugin repository last summer, in August of 2019, and they estimate is probably installed on less than 3,000 sites all total. Yet that accounted for more than half of all attacks. These guys were just - maybe they were recycling some old attack code, or they had it handy, or why not try this one, maybe we'll get lucky.

There was a cross-site vulnerability, cross-site scripting vulnerability in Blog Designer, patched last year. They estimate that no more than 1,000 vulnerable installations of that one remain, and it's been the target of previous campaigns. There was an options update vulnerability in WordPress's GDPR compliance tool, which was patched in late 2018, which would have allowed attackers to change the site's home URL in addition to other options.

Those are some powerful options. I think we talked about this at the time. And although this plugin, because it's WordPress's, has more than 100,000 installations, it was fixed two years ago, and they estimate now that today no more than 5,000 sites remain vulnerable to it. But still, I guess it's low-hanging fruit, just not a lot of it. There's also an options update vulnerability in something called Total Donations which would allow attackers to change the site's home URL, again an options update vulnerability. This was removed permanently from the Envato Marketplace that sells lots of add-on PHP apps, early last year. They estimate that less than a thousand total installations remain. But still, if you've got it, and you didn't update it, you could be a victim of it.

And, finally, there's a cross-site scripting vulnerability in the WordPress Newspaper theme, which was patched four years ago, in 2016. And it's also been targeted before, and now. So of course the takeaway here is obvious to all of us. Admins of WordPress sites must be responsible about updating their plugins and removing those that are no longer present in the WordPress repository. There's a reason they were yanked from that repository.

And if this hasn't gotten the attention of all WordPress admins yet, there's also a critical WordPress plugin, two plugins, present on over one million sites. Wordfence's threat intelligence team also reported last week that they detected a new campaign attempting

to abuse two other bugs in WordPress in ongoing attacks. The attacks are attempting to exploit security vulnerabilities in two things: Elementor Pro and something called Ultimate Addons for Elementor. The flaws potentially allow remote code execution of arbitrary code and a full compromise of unpatched targets.

On the 6th of May, which was what, last Wednesday, they wrote: "Our Threat Intelligence team received reports of active exploitation of vulnerabilities in two related plugins, Elementor Pro and Ultimate Addons for Elementor. We have reviewed the log files of compromised sites to confirm this activity." Remember, a million sites are using this thing. "As this is an active attack, we wanted to alert you so that you can take steps to protect your site. We are intentionally limiting the amount of information this post provides because this is an ongoing attack, and the most critical vulnerability has not yet been patched. There are two plugins" - oh, and they notified the Elementor people, didn't get a reply. "There are two plugins affected by this attack campaign."

They said: "The first is Elementor Pro, which is made by Elementor. This plugin has a zero-day vulnerability which is exploitable if users have open registration." So this is apparently involved in the registration loop of a site. And if your site allows people to register, and you're using Elementor Pro, whatever that does, that's not good. Then they updated this the next day, on May 7th, to say that: "Elementor has released version 2.9.4 of Elementor Pro. Our threat intelligence team has verified that this patches the vulnerability. We recommend updating to this version immediately." And of course we'll talk in a second about why that may not be enough.

"The second affected plugin," they write, "is Ultimate Addons for Elementor, which is made by Brainstorm Force. A vulnerability in this plugin allows the previous vulnerability, the Elementor Pro vulnerability, to be exploited, even if the site does not have user registration enabled." So if you were using Elementor Pro, weren't an open registration site, but you also had the Ultimate Addons for Elementor, the vulnerability in that one would allow you to bypass the closed registration on your site in order to get to the Elementor Pro vulnerability if you haven't yet patched it. They said: "We estimate that Elementor Pro is installed on over one million sites, and that Ultimate Addons has an install base of around 110,000." So these are big attack targets.

So they said: "To be clear, this does not impact the free Elementor plugin." Anyway, it goes on about critical severity. It's a remote code execution, full-site takeover, writing code, really ruining your month or more, and so on. At the time, prior to the update, they were recommending that, because Elementor Pro had not been yet patched, that people who needed it back down to the free version that was not vulnerable, use that until it's patched, then go back up to the Pro which you've presumably purchased. So anyway, if any of that rings a bell, make sure that you've got yourself patched because the 60 million WordPress sites, that's one out of every 60 WordPress sites was using this thing that was found being exploited when it came to their attention.

And of course the takeaway from all of this is somewhat more encompassing. Our software, all of our software, is becoming increasingly complex. And the development style which we can see from this podcast, which has evolved, is powerful because it's so modular. Third-party code libraries are obtained and integrated at the code level. Third-party plugins are obtained and integrated at the operational level. And in some cases, that resulting endpoint solution is further enhanced by an automation layer above that which pushes its buttons.

So the result is a massive, complex, and understandable only from a distance in overview system about which no true assertion of knowable security can be made because the system itself at the level of detail required to make any such assertion is unknowable, or at least unknown. And even if every modular component were itself secure, which history teaches us is never effectively true, each module makes assumptions about the nature of

its input and output. And those assumptions may not have been fully understood and appreciated by the developers who plugged all of the pieces together. As a result, the system will almost certainly be able to exhibit unexpected behavior when those assumptions are deliberately violated by attackers.

So what we've created with this ad hoc assemblage of disparate components is inherently insecure. But it's a choice we've made in the name of expediency. And, I mean, it's a huge gain. I mean, the power factor gain is not something I'm suggesting we can give up at this point. Because it is so seductively expedient. We obtain almost magical results when big systems like this are glued together. But what we don't get is robust security. It's just not available from this approach.

So we're left with doing the best we can. And that means monitoring and patching. And responsible patching requires knowing when there are patches. And knowing when there are patches requires maintaining a multitude of open lines of communication to the maintainers of every piece of this massive puzzle and then responding rationally with due speed when any significant new problem arises. So maintenance is the price we pay for expediency. And what we see is that all too often that price is not paid.

So anyway, there were just so many of these sorts of things we're seeing. I mean, hopefully no one listening to this podcast was among the one in 60 WordPress websites that were vulnerable, or they got the notification from Elementor Pro or of the Elementor Pro problem from Elementor and jumped on it fast before their site could be exploited. No one wants people to get hurt. People are being hurt.

Leo: I use WordPress.com. And I presume, you know, they vet the plugins that you can use, and I presume they also keep them safe and up to date. But that's a good question. I'll have to check. Because they really host it. It's a host-managed system.

Steve: I also have a WordPress blog.

Leo: Do you self-host, or do you do the WordPress.com managed hosting?

Steve: I self-host, although I'm linked in through Jetpack, which is their add-on.

Leo: Right.

Steve: And it provides a lot of the same features.

Leo: Jetpack's great, yeah.

Steve: Yeah. So it's sort of the compromise. It gives me Akismet in order to do spam blocking. And, boy, it blocks all kinds of junk. It also notifies me of updates. Sometimes it will notify me that it autonomously updated my WordPress installation. I've given it permission to do so. I'd rather that than to be exploited. I guess I'm of the opinion that the best solution is to be very sparse with the add-ons you use. And this is like how we feel about apps on our phone; right? If you just click, ooh, look at this candy store of free stuff, and you just load your device down with stuff, every one you add, the individual

opportunity for a problem might be small. But when you keep adding them, the effective opportunity increases.

So I have some mailing agent and maybe only - I have a very small number of additions. And I think actually most of them are from WordPress themselves, to add a couple features. I have like a Twitter feed thing and that kind of stuff. But very little.

Leo: Yeah, the less, the better. Especially with these - a lot of them are, I mean, that Elementor I think might be a little janky. You know what I'm saying?

Steve: They've made it onto a million sites, whatever it is.

Leo: Well, that's true. But that doesn't, you know, I mean, there's a lot of sites out there.

Steve: Yeah, true. Okay. So against that somewhat dispiriting background, we've got another example of a web-based security emergency: vBulletin.

Leo: Another great safe PHP project.

Steve: Exactly. It's widely used Internet forum software which currently powers more than 100,000 websites around the world. Written in PHP with a SQL backend, it's in use by many of the Fortune 500 and other major companies. It has the mixed blessing of being old, and old is not always good. Several of that system's early developers had a falling out with management and left vBulletin to form XenForo, which our listeners all know...

Leo: Is what you use, yeah, yeah.

Steve: ...is what I chose. They had the opportunity, and for something like this I recommend it, of just starting over from scratch. Use everything you learned during the first implementation. I think this is actually their fourth overall. And just say, okay, now we think we know how to do it. You'll find out you don't, you know, and that you kind of have to fudge the details and things. But that's the nature of it. And then maybe the next time you'll know how to do it.

vBulletin remains a very active going concern, and the maintainers of vBulletin just announced a CRITICAL patch, in all caps, without revealing any information, not that that helps a lot, or details about the underlying security vulnerability. It's only identified by its CVE tracking designator of 2020-12720. But the word on the street is that anyone running vBulletin must update immediately to the latest release of their major release track. Because vBulletin remains quite popular on the web and because being written in PHP it is effectively running its own source code. So it's one of the hackers' favorite targets.

The maintainers are clearly hoping that withholding the details of the flaw could buy some time for their users to apply the patch before hackers can reverse-engineer the fix and use that to exploit those sites which have not yet updated. However, as has occurred previously, researchers and hackers both have already started reverse engineering the

patch to locate and understand the vulnerability. The national vulnerability database has analyzed the flaw and revealed that it is indeed CRITICAL, in all caps, and originating from an incorrect access control issue.

vBulletin's bulletin said: "If you are using a version of vBulletin 5 Connect prior to 5.5.2, it is imperative that you upgrade as soon as possible." Nothing's been said about whether vBulletin version 3 and version 4 - as I noted, it's been around, I think it was '07. No, no, '04, I think. So it's been around for quite a while. And if they have the problem, that's also really bad since those versions are combined about equal to the number of v5 installations. In other words, there are people foolishly running version 3 and version 4, for which there has been no maintenance for a long time. Still out on the Internet, presumably, we'll hope, no major corporations.

And of course, Leo, as I mentioned to our listeners at the time, that's why before I was willing to bring up a PHP-based bulletin board, I put a physical firewall between that separate machine and the rest of my network. And that firewall only allows a specially designated bit of traffic out to the outside world, and none to the rest of my network because, I mean, this stuff is, as I said, these systems are so cool in what you're able to glue together and do. But with that expediency comes a price.

So this is an unfortunate thing that we see with this sort of software, just as with Microsoft finally refusing to maintain Windows 7, despite its still massive install base, the vendors of complex web forum software occasionally produce a major upgrade that requires effort on the part of all their sites to make the change. I've already done that once with the XenForo forums that I maintain. I had to take it all down, back it all up, do a major update, have the database format converted and all that. I mean, it's easier just to say, eh. But you just can't. Maintaining that kind of software comes with a responsibility. So it's truly a mess.

And last September, we talked about this at the time, something very similar to this occurred with vBulletin when a remote code execution vulnerability was discovered and patched, and sites were urged to upgrade immediately. Many didn't, and many were successfully attacked because the attackers wasted zero time in pouncing. This is why users of these systems, admins must be able to dedicate the time of someone who is just as committed to patching the systems as the attackers are to attacking it. It's just not something you can take for granted. Currently there's no proof of concept code available, and no indication that the new vulnerability is being exploited in the wild. But if history is any guide, we may be talking about that next week.

The clock is ticking because Charles Fol, a security engineer at Ambionics, who was the confirmed discoverer of the vulnerability, who also responsibly reported it to the vBulletin team so they could create their updates, has stated that he plans to release more information during the upcoming French SSTIC security conference scheduled for the 3rd through the 5th of June, thus early next month, about three weeks from tomorrow. And so anyone running vBulletin, you want to download and install the patch for your version. 5.6.1 needs to patch to level one. 5.6.0 same, and 5.5.6.

So this thing's been diverging a little bit. There's three different patches you need to apply, depending on which one you're using. And notice, no hope for version 3 and version 4 people, even though I don't know that they're affected by this. Hopefully they're not. I mean, maybe they're just stable, and they're happy with what they've got.

And Leo, for the first time ever, for the first time ever, I might actually considered attending this summer's Black Hat and DEF CON security conferences.

Leo: Oh, Steve. Hadn't you heard?

Steve: And in fact I would encourage all of our listeners to do so. They've been confirmed now to be going online.

Leo: This is great, actually.

Steve: Yeah.

Leo: All of these conferences, suddenly you can attend. It's great.

Steve: Well, and you don't have to leave your tech at home or lock it up in an RFID-proof enclosure.

Leo: Yeah, yeah, it's safe. Right.

Steve: The two conferences, well, as we know, they are the industry's two biggest cybersecurity conferences annually. But of course for the first time ever they will not be held in Las Vegas. Thanks to the coronavirus, they'll be going virtual. The two conferences were initially scheduled to take place in Vegas, as always, sequentially, back to back, during the first two weeks of August. Black Hat would have been August 1st through 6th, and DEF CON 7th, 8th, and 9th. But of course they're following in the footsteps of many other cybersecurity conferences and non-cybersecurity conferences, all conferences, that have necessarily switched into the cyber world for the purpose of discussing cybersecurity. We don't have many details at the moment.

But we do know that both conferences are planned to live stream the talks to their paying attendees. And normally there are parallel tracks. And so like they've got multiple talks going on at the same time. I don't know how they'll handle that. Maybe they'll just do them that way. Or maybe they'll make them serial. They could certainly extend the hours. Anyway, we'll see what happens. Since no changes in dates have been announced, both conferences are expected to still take place during their previously announced dates, although all attendees can cancel their airline tickets. I'm sure that's already happened.

The Black Hat team has only posted a short statement, but Jeff Moss, DEF CON's manager, went into some detail about what led up to his decision to go virtual. He wrote: "While I made the decision to cancel the in-person conference a month ago on April 11th, the delay in announcing was been due to learning how to actually cancel," since he'd never had to before.

Leo: It's more complicated than you think because they prebook the venue and all sorts of stuff; right?

Steve: Yes, exactly. He said: "It has taken weeks of working with staff, lawyers, accountants, and Caesars Palace." He says: "I didn't want to endanger the future of the conference by tweeting that we were canceling before we understood and were confident we could navigate the process." So going forward, Jeff has said that DEF CON will continue to be an in-person event. That is, this is not the straw that said, oh, why are we all meeting in person? And I really get that. I mean, a huge amount of the charm of

Black Hat and DEF CON is being there physically. So next year DEF CON 29 is still scheduled to be an in-person event, August 5th through 8th of 2021. But not this year. This year you get to tune in and do it from the comfort. And I may attend. I'm like, why not?

So Samsung has patched a critical bug affecting the past six years of smartphones. Since 2014 the Android OS flavor running on Samsung devices has included a handler for the custom Qmage, which was a new one for me, Q-M-A-G-E, the Qmage image format that has the extension .qmg. A researcher with Google's Project Zero discovered a way to exploit how Skia, S-K-I-A, which is Android's graphics library, handles these Qmage images sent to a device. And this can be exploited as a zero-click attack through the reception of MMS messages without any user interaction. The vulnerability occurs because Android redirects all images sent to a device through this Skia library for processing, such as generating thumbnail previews.

Google's researcher developed a proof-of-concept demo exploiting the bug against the Samsung messages app, which is included in all Samsung devices and of course is the handler of all incoming SMS and MMS messages. And outgoing, too. I mean, that's the messaging app. The bug was exploited by sending repeated MMS messages to a Samsung device. Each message attempted to guess the position of the Skia library within the Android phone's memory space, which is required in order to brute force Android's ASLR, the Address Space Layout Randomization protection.

Once the Skia library has been located in memory, a single final MMS message delivers the actual Qmage payload, which executes the attacker's code on a device. So the attack typically requires between 50 and 300 MMS messages, you know, basically you're guessing where Skia is located in memory, so it's going to be a range until you guess right. So it typically requires between 50 and 300 messages to probe and bypass ASLR, essentially by brute force. And that usually takes around 100 minutes on average. So while the attack might seem noisy, it can be improved to execute without any alert to the user.

The researcher at Google wrote: "I have found ways to get MMS messages fully processed without triggering a notification sound on Android, so fully stealth attacks might be possible." The vulnerability was discovered back in February and reported to Samsung, who then patched the bug in this month's May 2020 security updates. And of course thanks to ASLR, this is a perfect use case for it, the attack is functionally limited to targeted attacks. Were it not the case, if you could just send out MMS messages and immediately perform a remote device compromise, that'd be really bad. So it's a good thing we have ASLR here.

Anyway, if you or someone you know are a Samsung smartphone user who might be a target of a targeted attack - again, it's not expected to just be sprayed because you can't spray this one - be certain to obtain this month's Samsung patches. Hopefully you've got a phone that is still current and patchable, even if this Qmage format was added in 2014. No other smartphones from other vendors appear to be impacted because only Samsung has modified their version of the Android OS to incorporate this custom Qmage image format which was developed by a South Korean company, Quramsoft.

So anyway, just a note for any Samsung users. Hopefully you're on the patch cycle, and you got the patch. And it wasn't found. It's not a zero-day. It wasn't known to be exploited in the wild. So this is just one of those things where Google Project Zero comes to the rescue again, ahead of it being a zero-day. And Leo.

Leo: Yes.

Steve: Zoom purchased Keybase.

Leo: I know. I never did get you to try it, did I, Steve.

Steve: No.

Leo: Too late now. Well, I guess you could still try it. They didn't kill it. But it doesn't look good.

Steve: Well, they effectively killed it because they bought the brain trust.

Leo: Right.

Steve: And the reputation, which is what they wanted. Last Thursday, May 7th, Zoom's CEO Eric Yuan blogged the news of Zoom's latest move for furthering the security of their wildly successful, thanks to the coronavirus, teleconferencing platform. And in the URL, I got a kick out of the fact that it's got WordPress in it. It's blog.zoom.us/wordpress/ blah blah blah.

Leo: And we guess we know where Zoom's blog is.

Steve: Exactly. The title of Eric's posting was, and there are some technical goodies in here, so I'm going to share it with our listeners: "Zoom Acquires Keybase and Announces Goal of Developing the Most Broadly Used Enterprise End-to-End Encryption Offering." He says: "We are proud to announce the acquisition of Keybase, another milestone in Zoom's 90-day plan to further strengthen the security of our video conferencing platforms. Since its launch in 2014, Keybase's team of exceptional engineers has built a secure messaging and file- sharing service leveraging their deep encryption and security expertise. We are excited to integrate Keybase's team into the Zoom family to help us build end-to-end encryption that can reach current Zoom scalability.

"This acquisition marks," he says, "a key step for Zoom as we attempt to accomplish the creation of a truly private video communications platform that can scale to hundreds of millions of participants, while also having the flexibility to support Zoom's wide variety of uses. Our goal is to provide the most privacy possible for every use case, while also balancing the needs of our users and our commitment to preventing harmful behavior on our platform. Keybase's experienced team will be a critical part of this mission."

He says: "Today, audio and video content flowing between Zoom clients, for example, Zoom Rooms, laptop computers, and smartphones running the Zoom app - is encrypted at each sending client device. It is not decrypted until it reaches the recipients' devices. With the recent Zoom 5.0 release, Zoom clients now support encrypting content using industry-standard AES-GCM" - we've talked about this previously - with 256-bit keys. However, the encryption keys for each meeting are generated by Zoom's servers. Additionally, some features that are widely used by Zoom clients, such as support for attendees to call into a phone bridge or use in-room meeting systems offered by other companies, will always require Zoom to keep some encryption keys in the cloud. However, for hosts who seek to prioritize privacy over compatibility, we will create a new solution.

"Zoom will offer an end-to-end encrypted meeting mode to all paid accounts." So that'll be a value-add for payment, for being a paid account. "Logged-in users will generate public cryptographic identities that are stored in a repository on Zoom's network and can be used to establish trust relationships between meeting attendees." And of course all of our podcast listeners who have followed along about public key crypto know how that'll work. That's essentially meaning that Zoom will be a database manager for all logged-in users' public keys. And that will allow them to synthesize a key that only certain people can get and vice versa.

So he continues, basically establishing trust relationships between meeting attendees. He says: "An ephemeral per-meeting symmetric key will be generated by the meeting host. This key will be distributed between clients, enveloped with the asymmetric key pairs, and rotated when there are significant changes to the list of attendees. The cryptographic secrets will be under the control of the host, and the host's client software will decide what devices are allowed to receive meeting keys, and thereby join the meeting. We are also investigating mechanisms that would allow enterprise users to provide additional levels of authentication.

"These end-to-end encrypted meetings will not support phone bridges, cloud recording, or non-Zoom conference room systems. Zoom Rooms and Zoom Phone participants will be able to attend if explicitly allowed by the host. Encryption keys will be tightly controlled by the host, who will admit attendees. We believe this will provide equivalent or better security than existing consumer end-to-end encrypted messaging platforms, but with the video quality and scale that has made Zoom the choice of over 300 million daily meeting participants, including those at some of the world's largest enterprises." And he goes on, blah blah blah. But basically that's the gist of it.

So they're essentially creating, using Keybase's reputation, and certainly their technology, I'm sure, that that advisory panel said to Eric, you know, just go buy the practical implementation crypto knowledge that you need because you're in a hurry. So just go buy it. And that's what Zoom did. And I think it was wise. It gives them headlines. Clearly they're maintaining this momentum they have of rapidly moving forward to dispel concerns about Zoom's security platform and infrastructure, and actually offering some new features. The idea, you know, this is clearly focused on a host-centric, a host control of who gets to participate, which makes a lot of sense. So anyway, again, I've said several times, bravo to Zoom, and that the steps they're taking would make great material for a business management course in business school. And bravo.

Leo: Yeah. I'm going to miss them because I use them for a variety of services, and they're not easily replaced. But...

Steve: That's Keybase, yes.

Leo: Yeah, Keybase. But I think Zoom will benefit from it. Keybase does not use...

Steve: Well, it was all open source and GitHub; right?

Leo: Not all of it. Their server side was closed source. But yeah, there's enough open source stuff that somebody could fork it, and I hope they do. Keybase did not do - initially did what Eric described. They would host your PGP or GPG public key in a centralized key database, much like the key servers do, yeah.

Steve: Right.

Leo: But really their most interesting thing was something that they eventually turned to, which is device authentication. So that you would use an existing Keybase device to add a new Keybase device, and there was a chain of trust going through the devices. It was very different than the public key/private key crypto system that they used. I thought that was most interesting. And I wonder if Zoom will do something like that. But we'll see. These guys are very good, I think very talented crypto guys. So, I mean, they're certainly an asset to Zoom. And we can find other things to replace Keybase.

Steve: And certainly having developed and perfected that interdevice chain of trust, they would be able to offer it to Eric and say, hey, you know, we've got this stuff.

Leo: This is a better way of doing it, yeah.

Steve: Yeah.

Leo: Much better way of doing it. So anyway, we'll see, yeah.

Steve: Yeah, very cool. So since last week my work on SpinRite has been focused on implementing an updated boot prep system because we're going to need it. We're going to need it soon for the existing testers. And I'm going to want our podcast listeners to have easy access to the SpinTest, the new technology R&D side, in order to make sure it works on all of their stuff. So that's going to require that it is easily able to create a bootable thing. I wrote the current SpinRite version 6, SpinRite 6 Windows boot media prep system 16 years ago, back in 2004. And I was a little taken aback, because I'm back into the source, when I'm looking at code for compatibility with operation on Windows 95 and 98.

Leo: Oh, it's been a while, hasn't it, Steve.

Steve: Yes, it has. It's got that. If you need it, we've got that. And after all, at the time, Windows 98 was only six years old. So it was still a going concern. And at the time the most reliable boot medium was the 1.44MB so-called double-density, because you could have 720K if you didn't have the notch in your disk, the double-density floppy disk. And of course CDs were in second place. Back then, not all systems could even boot from USB. That was an emergent property of BIOSes. But as all SpinRite users now know, well, know then, I supported all of those modes of creating bootable media.

The largest USB thumb drive capacity numbered in the hundreds of megabytes. Not gigabytes, megabytes. So I remember having a 512MB thumb drive, it's like, oh, I'm never going to fill that up. That was, again, 2004. So the old-school cylinder head and sector, so-called CHS boot sector, which was able to handle drives of up to 8GB - nobody had a thumb drive of that size - seemed entirely adequate at the time. Of course, if history is our teacher, we know the rest. That certainly is the case no longer. And what's more, if SpinRite is to have a future, and I'm committed to that future, it will need to be

able to boot on either older BIOS-based hardware or UEFI systems, without either a classic BIOS or DOS compatibility.

So I'm in the process of building a new boot media prep system that'll give me control over all aspects of the drives it prepares. For now, that just means that any USB thumb drive it's asked to prepare, regardless of size, will be easily made bootable on any BIOS and DOS system. And in the future we'll be ready to add UEFI and operation without any DOS OS at all. And speaking of UEFI, one of the things that has been brought to my attention during this return to work, as I'm headed toward a new release of SpinRite, is the growing importance of UEFI. Intel no longer supports the traditional BIOS at all, and Apple has dropped the so-called CSM, that's the Compatibility Support Module, from their more recent offerings.

So this all suggests that the BIOS's days are numbered, and that number may not be very big. As everyone knows, my plan has been to produce a series of 6.x releases to bring SpinRite up to the latest hardware standards. The idea was for that to buy me time to start over from scratch on the complete reconceptualization of SpinRite as a drive and a file system and a file-aware fully multitasking data maintenance and recovery tool. So, for example, you could SpinRite everything at once, for example, which is completely doable. But SpinRite's current user interface has no possibility of that. And I'm really looking forward to doing that.

But with the BIOS disappearing more quickly than I wish it were, I'm worried that SpinRite's near future users, and even recent Mac purchasers, may again be left without a way to run SpinRite. So what I'm thinking about is that perhaps SpinRite 7 should happen immediately after SpinRite 6, after the SpinRite 6 series, to address this disappearing BIOS and DOS problem by adding native UEFI booting and the ability to run on the machine without any underlying OS. DOS, FreeDOS, any DOS is not compatible with UEFI and never apparently will be. The FreeDOS people have explicitly stated: We couldn't care less. We have no intention of supporting UEFI.

Well, okay. The good news is I don't use DOS for much of anything. Memory allocation, but that's one of the reasons I just wrote my own memory allocator from scratch for this next round of work. I'm not using DOS's. I use it to print to the screen because, well, actually some of the time. I do a lot of that myself in that whole multitasking user interface. So it won't be very difficult to say, okay, I don't need DOS at all. Mostly it just loads the code, and then I tell it to get out of the way. And I'm obviously going to need that technology, that is, UEFI and no depending upon a nonexistent-in-the-future operating system, for everything that follows anyway. So doing that first is not a diversion, it just moves things around in sequence to keep a SpinRite that I can create now rather than some time in the future.

And of course we all know I'm not typically very quick with these things. Look at SQRL. Yeah, I got it done, but it took longer than I expected. And that gets us a SpinRite that ought to be able to span the time until SpinRite v8 is ready. And of course, yeah, I love the name SpinRite v8. That's sort of appropriate. So anyway, those are my plans at the moment. So I'll keep thinking about it, and I'll firm them up as we go. In the meantime, I'm well at work on SpinRite 6 and hope to have something for our listeners to play with before long.

Leo: What year will v8 come out?

Steve: So the one mistake I always make is I overpromise, and then I get stuck behind the, dare I say the eight-ball. I'm not going to let that...

Leo: Yeah. Are you going to skip 7?

Steve: No. The idea will be I will do 7 immediately in order to give SpinRite UEFI support, which means the ability to run without DOS. Because SpinRite users are going to need that, even if it's only able to run on one drive at a time. But, boy, Leo, is it fast. Oh, my lord. I saw it run on a drive, that is, the current testing version, 258 megabytes per second, which says, what, a gig every four seconds. So it suddenly becomes practical again to use SpinRite to maintain drives. So I'm very...

Leo: These drives have gotten so big, yeah.

Steve: ...very excited about it, yeah.

Leo: Good. Well, don't feel the pressure to get to 8 too quickly. You've got time.

Steve: Well, I mean, I'm loving development. I'm having such a ball being back working on SpinRite again. And yeah, I mean, I want to enjoy the process and just be working on SpinRite 8. That's what I'll be doing. And I'm thinking that I'm going to take a different development approach, which is rather than holding everything back until I have anything, just releasing what I have as I go because it'll be fun to play with, and probably be incrementally useful. So anyway...

Leo: Nice. When's the VR version coming out? Just teasing. Just kidding.

Steve: Just so you can look around and see your drive in the sky and pull a file out of the air. Oh, I was looking for this document. Thank you for recovering it for me.

Leo: Some day we won't even have files anymore.

Steve: Here's my lost dissertation, yes. Thunderspy. Okay. So first, here's the mixed-blessing summary written by Bjrn Ruytenberg - sorry, best I can do - from the Eindhoven University of Technology, describing what his research has uncovered. He wrote: "Thunderspy targets devices with a Thunderbolt port." And as we know, that's a USB Type C port, which is Thunderbolt-enabled, typically. "If your computer has such a port, an attacker who gets brief physical access to it can read and copy all of your data, even if your drive is encrypted and your computer is locked or set to sleep." Now, that sounds really bad, until he gets around to the part about the screwdriver, which we'll get to in a second. But it turns out that's not really necessary, either.

He says: "Thunderspy is stealth, meaning that you cannot find any traces of the attack. It does not require your involvement, so there's no phishing link or malicious piece of hardware that the attacker tricks you into using. Thunderspy works even if you follow best security practices by locking or suspending your computer when leaving briefly, and if your system administrator has set up the device with Secure Boot, strong BIOS and operating system account passwords, and enabled full disk encryption. All the attacker needs is five minutes alone with the computer, a screwdriver, and some easily portable hardware." It turns out screwdriver, not so necessary. But to do the full seven exploits you do need a screwdriver.

He says: "We have found seven vulnerabilities in Intel's design and developed nine realistic scenarios how these could be exploited by a malicious party to get access to your system, past the defenses that Intel had set up for your protection. We've developed a free and open source tool, Spycheck, to determine whether your system is vulnerable. If it is found to be vulnerable, Spycheck will guide you to recommendations on how to help protect your system."

And I got so involved in his paper that I forgot to put a link. I think it's, oh, it's Thunderspy.io. So that's easy to find: https://thunderspy.io. And there you'll find a description, and he's got Windows and a Linux version of his tool and some other stuff. Okay. So to get a quick sense for what this means, what he's talking about - oh, and I should also mention, you know, this has been described as an "Evil Maid" attack, meaning that the physical presence thing is what we're now calling the Evil Maid. And I would argue that it's not really an Evil Maid unless your Maid was trained at MIT.

Leo: That's an excellent point. That's an evil spy. Evil, yeah, yeah.

Steve: Yeah. So to get a quick sense for what this all means, the next step is to look at what he wrote in the abstract for his formal security research paper. It was 23 pages, I think, long. And I'm not going to drag everyone through it. It's not necessary because we want to get the gist of this. But it provides, the abstract provides some necessary and interesting background on Thunderbolt and what that implies. So he explains: "Thunderbolt is a proprietary I/O protocol promoted by Intel and included in a number of laptops, desktops, and other systems. As an external" - and here's the key. "As an external interconnect, it allows exposing the system's internal PCI Express (PCIe) domain to external devices." This is where we insert "What could possibly go wrong?"

He says: "This enables high-bandwidth, low-latency use cases, such as external graphics cards. Being PCIe-based, Thunderbolt devices possess Direct Memory Access-enabled I/O, allowing complete access to the state of a PC and the ability to read and write all of system memory. In recent years, the former characteristic - the ability to read and write all of system memory - has prompted research into attacks collectively known as 'Evil Maid,' which require an attacker-controlled device and only seconds of physical access to the computer.

"Industry response has been twofold. First, hardware and OS vendors incorporated support for DMA remapping using I/O Memory Management Units (IOMMUs), which imposes memory protections on DMA. However, following various implementation issues, OS vendors classified DMA remapping as an optional countermeasure requiring driver support. Second, revised Thunderbolt controllers introduced a software-based access control measure enabling users to authorize trusted devices only." And I heard you mentioning on MacBreak Weekly, Leo, the notion of a whitelist. Unfortunately, we'll see that doesn't actually provide much protection, in fact.

He says: "As a result, unidentified devices should be barred from system access without prior user interaction." Sounds great, although we're not supposed to call it "whitelist" anymore, I realize. I've forgotten what it's going to be.

Leo: Allow list. An allow list, yes.

Steve: An allow list. Okay.

Leo: Shocking.

Steve: "In the context," he's saying, "of Thunderbolt, studies have primarily focused on employing DMA and IOMMU attacks on the PCIe level. We therefore investigate the feasibility of breaking Thunderbolt protocol security by analyzing the protocol and its software and hardware stack, as well as associated PCIe-based technology. In our study, we have found and experimentally confirmed multiple vulnerabilities that break all primary Thunderbolt 3 security claims. Based on our ongoing research, in this report we disclose the following vulnerabilities." Seven of them. "Inadequate firmware verification schemes, weak device authentication scheme, use of unauthenticated device metadata, backwards compatibility with legacy protocol versions, use of unauthenticated controller configurations, SPI flash interface deficiencies, and no Thunderbolt security on Boot Camp.

"Finally," he says, "we present nine practical exploitation scenarios. Given an Evil Maid threat model and varying security levels, we demonstrate the ability to create arbitrary Thunderbolt device identities; clone user-authorized Thunderbolt devices; and, finally, obtain PCIe connectivity to perform DMA attacks. In addition, we show unauthenticated overriding of security level configurations, including the ability to disable Thunderbolt security entirely, and restoring Thunderbolt connectivity if the system is restricted to exclusively passing through USB and/or DisplayPort." That is, if Thunderbolt protocol has been disabled on the USB port, he can get it back. "We conclude this report by demonstrating the ability to permanently disable Thunderbolt security and block all future firmware updates." In other words, once in, they can stay in.

So of course we've used the very useful term "security perimeter" many times before. It's a conceptually clean way of establishing the idea of what's under protection, where the barrier to penetration is, what sort of protection is available and needed, and where the resulting vulnerabilities lie. So if nothing else, it seems very clear that this was necessary research, even if its theoretical exploitability seems low.

And frankly, as we'll see, it's actually not very low. But it is not remote in any way. It does require local access. It seems very clear that this was necessary research. Even if it's theoretical at this point, just look at how the purely and equally theoretical, actually way more theoretical, this is actually exploitable, Spectre and Meltdown vulnerabilities hugely deepened our understanding of the exploitability of the many once-believed-to-be-safe CPU performance optimizations that all of our chips had at the time.

So this research, the nature, this kind of research is super useful stuff. And there appears to be a real desire to export a system's internal bus through an easy-to-use serial connector. This dates back to the original 1394 Firewire, where exploits of its exported hardware bus similarly afflicted that interface. So here again it appears that, just as with Firewire, security was layered on later, almost as an afterthought. I mean, remember they were talking about adding DMA protection. What do you mean, adding? How could you not have it, like from the beginning? Oh, yeah, we're going to put the PCIe bus, your internal system bus, on a connector on the side of your laptop or on the back of your PC. How does that sound? Think of all the things you can connect to it. Yes. And bad guys.

So as we observed a decade ago on this podcast, if the bad guys have access to the hardware, hardware of any kind - 10 years ago we were talking about a DVD player which must fundamentally be able to decrypt the disks, which are encrypted, in order to play them. If the bad guys can get to the DVD player, they can get the decryption keys. And similarly, no matter what you do, you cannot protect things locally. But you sure can make it easier to exploit, which is what a pluggable PCIe bus does.

So here in our present situation, if you have an MIT-trained Evil Maid with unfettered access to a system whose hardware is exporting its internal bus to the outside world, then yeah, once again all bets are off. The research paper, as I mentioned before, is 23 pages of detailed "here's how I did it" information. But a summary of the seven vulnerabilities is not overly long, and it'll give us a better sense for Thunderbolt's measures and countermeasures. So I've got seven - this is an explosion of those seven points.

First, inadequate firmware verification schemes. "Thunderbolt host and device controllers operate using updatable firmware stored in its SPI flash." That's Serial Programming Interface, SPI, which is the way all those little itty-bitty eight-pin chips on motherboards now operate is serially. Everything's gone serial because we can do that now speedy. And it just completely reduced our pin count. So we don't need parallel buses, we just have serial buses. So it's been a big win.

So "Thunderbolt host and device controllers operate using updatable firmware stored in its SPI flash. Using this feature, Thunderbolt hardware vendors occasionally provide firmware updates online to address product issues after release. To ensure firmware authenticity, upon writing the image to the flash, Thunderbolt controllers verify the firmware's embedded signature against Intel's public key stored in silicon." Sounds great; right?

"However, we have found authenticity is not verified at boot time, upon connecting the device, or ever again, at any later point. During our experiments, using a SPI programmer, we have written arbitrary unsigned firmware directly onto the SPI flash. Subsequently, we've been able to verify Thunderbolt controller operation using our modified firmware." So in other words, the sanctioned way of writing the Thunderbolt controller firmware verifies the firmware's signature, but only at the time of that writing. So anything that bypasses the sanctioned means of updating the firmware can make any changes it wishes. And the then-broken firmware will henceforth be rerun without ever being reverified.

And since the firmware is the thing that updates itself, you can disable the firmware updating, make it look like it worked, but never actually update the firmware. So that would require intimate knowledge of the system's motherboard, whether it's desktop or laptop. You would need to attach a little SPI programmer clip to the back to the SPI chip or to the controller's SPI programming pins, if it exports those. I mean, we know it does because it needs to be reprogrammed. And that's where you need somebody really trained up for this particular problem.

But the point is that the firmware is only verified once it's written through the official channel and never subsequently. So as, Leo, you were saying on MacBreak Weekly, and I completely agree, this is the kind of thing that has the NSA and the CIA just salivating because, if they didn't already know this, they know it now. And this is their scale of attack. But there are way easier ways to get up to some mischief.

Number two, weak device authentication: "As noted above, device identification" - and this is before these points were laid out. They talked about how it's just some short strings of metadata, basically. So they say: "As noted above, device authentication comprises several strings and numerical identifiers. However, we have found none of the identifiers are linked to the Thunderbolt controller or one another, cryptographically or otherwise."

In other words, this means that impersonation of any previously authenticated device is trivial. You briefly connect an identity capture device to anything that was previously permitted to connect, like a display or a graphics card or whatever Thunderbolt thing that's been whitelisted. You read and capture the device's stated identity and simply

impersonate that device to obtain full and unfettered access yourself. And there's, again, no authentication beyond just an unencrypted, unauthenticated string.

Number three, which is sort of a repetition of two, use of unauthenticated device metadata. He says: "Thunderbolt controllers store device metadata in a firmware section referred to as Device ROM (DROM). We have found that DROM is not cryptographically verified. Following from the first issue, this vulnerability enables constructing forged Thunderbolt device identities. In addition, when combined with the second issue" - which is lack of authentication of stated identities - "forged identities may partially or fully comprise arbitrary data." So in other words, even if a previously authorized device is not available for identity cloning, it's possible to simply plant a malicious device's identity metadata into the system's Device ROM to give it then full access permission.

Point four, and this is always a bugaboo, backwards compatibility: "Thunderbolt 3 host controllers support Thunderbolt 2 device connectivity, irrespective of Security Levels." Which is a concept introduced in Thunderbolt 3. Well, since Thunderbolt 2 devices didn't know about that, if you plug in a Thunderbolt 2 device, well, we'll just not have any Security Levels.

"This backwards compatibility," they write, "subjects Thunderbolt 3-equipped systems to the vulnerabilities introduced by Thunderbolt 2 hardware." And I didn't do any digging to determine what those vulnerabilities in Thunderbolt 2 were, but presumably there were some. And as a consequence of the need for backward compatibility, they are subject to returning when you plug a Thunderbolt 2 device into your brand new spiffy Thunderbolt 3 system.

Problem five, the use of unauthenticated controller configurations. He writes: "In UEFI, users may choose to employ a Security Level" - that's capital "S," capital "L," that's a formally defined thing, there are four levels - "different from the default level. In storing Security Level state, we've determined that Thunderbolt employs two state machines, with one instance being present in UEFI, and another residing in host controller firmware.

"However, we have found firmware configuration authenticity is not verified at boot time, upon resuming from sleep, or at any later point. In addition, we've found these state machines may be subjected to desynchronization, with controller firmware overriding UEFI state without being reflected in the latter. As such, this vulnerability subjects the Thunderbolt host controller to unauthenticated, covert overriding of Security Level configuration." So again, it's clear that the security of the system could have been made much tighter, but it wasn't.

Six, SPI flash interface deficiencies: "Thunderbolt systems rely on SPI flash to store controller firmware" - the first vulnerability - "and maintain their Security Level state" - the fifth vulnerability. "In our study, we have found Thunderbolt controllers lack handling hardware error conditions when interacting with flash devices. Specifically, we've determined that enabling flash write protection first prevents changing the Security Level configuration in UEFI, again without being reflected in the latter; and, two, prevents controller firmware from being updated, without such failures being reflected in Thunderbolt firmware update applications. As such, when combined with the fifth issue, this vulnerability allows for the cover and permanent disabling of Thunderbolt security and will also silently block all future firmware updates."

And seventh, last but not least, no Thunderbolt security on Boot Camp. Meaning none. So all of these other problems are still affecting Apple devices that offer Thunderbolt ports. But on top of that, any security attempts that are there are relinquished: "Apple supports running Windows on Mac systems using the Boot Camp utility. Aside from Windows, this utility may also be used to install Linux. When running either operating system, Mac UEFI disables all Thunderbolt security by employing the Security Level

'None.' This vulnerability subjects the Mac system to trivial Thunderbolt-based DMA attacks." When running either operating system. So that also assumes that you could use Boot Camp to get to there, and that would set Thunderbolt security to none.

Leo: I don't think that's the case.

Steve: And then make it subject to attack. So you probably don't actually even need to boot those OSes. Again, I didn't dig into this deeply.

Leo: Yeah.

Steve: So where does that leave us? The complete lack of hardware-level enforcement of per-boot firmware verification means that our current hardware systems cannot, our current hardware systems, the stuff we all have now, cannot be made invulnerable to the "Evil MIT Grad" attack. Since the SPI chip holding the Thunderbolt firmware is on the system's motherboard, and since SPI programmers are a dime a dozen - literally, they're almost free on Amazon or eBay - if someone with sufficient knowledge were to gain access to a machine, they could override any protections.

So again, this is a heyday for any situation where it's possible to open up a device, have sufficient knowledge and planning ahead of time, knowing the make and model and so forth, you could figure out ahead of time where the SPI chip is and be ready to just clip it on, quickly reprogram it, unclip it, close the machine back up, and you now have future external access to the internal state of that machine through its Thunderbolt port.

But the more worrisome attack, to my mind, since it's something that any old untrained Evil Maid could do, even a naive maid, when an authorized external Thunderbolt device is available, it's simple to unplug that device, plug it into a Thunderbolt device ID cloning tool, grab its identity, then turn around and impersonate that device and obtain full access to the system's internal PCIe bus. It's just there. It's built in. So while it would be nice to think that this could be fixed in the future, the fact that device identity metadata is not currently authenticated means that impersonation is trivial.

And even if some future Thunderbolt 4 were to fix this, as it seems likely to try to because, after all, all of our Intel chips are running more slowly now because of Spectre and Meltdown, even though no one actually created an attack that worked, presumably there will be a response to this detailed takedown of Thunderbolt. If you are going to maintain backward compatibility to our current unauthenticated devices, I don't see how you ever bypass the impersonation attack. So backward compatibility bites us, unfortunately.

So our ultimate takeaway is that, by just about any definition of security, exporting a system's internal bus is almost always a really bad idea. And that's the case here once again. Very cool to have, but you really do need to plug it with epoxy if you don't ever want anybody to actually be able to plug into it.

Leo: Right.

Steve: Or electrically disconnect it. I guess maybe that'll be what happens is that there'll be a physical - well, on the other hand, if the bad guys get inside, they can just

reconnect it. And you would think, oh, I'm safe, my port's disconnected, when in fact it wouldn't be. So wow.

Leo: Well, there you have it, ladies and gentlemen.

Steve: Do not export your PCI bus.

Leo: But Apple does seem to have a way to keep it secure. I wonder what they're doing to keep it secure. In order to - insecure, you have to be running Windows or Linux under Boot Camp. That doesn't say that if you're running...

Steve: No, no. No, that's just this Security Level thing.

Leo: That setting is persistent after you exit?

Steve: No, it's a Security Level, and vulnerability six bypasses that.

Leo: But only when you're running Linux in Windows.

Steve: No, no. It's complicated. So there's this notion of four levels of security. And if you're using Boot Camp, then the Mac sets it to none.

Leo: Right.

Steve: But if you weren't, then you use the previous one through six vulnerabilities, which all Macs also have.

Leo: So I get it. I guess. Apple says they're not insecure. So I'll have to look more into this. Apple says they're not vulnerable to these other ones. But maybe I'm misunderstanding.

Steve: They have to be, well, I mean, I should look into it, too, because the device ID cloning is trivial, and there's no way they're not vulnerable to that. So, I mean, technically it's not their fault. I don't mean to be blaming Apple.

Leo: No, it's Thunderbolt, I understand, yeah, yeah.

Steve: Yeah, it's absolutely Intel's boneheaded idea of, like, wow, wouldn't it be cool to export the PCI...

Leo: Intel's kernel protection technologies do not bypass one through four, do not protect you against one through four?

Steve: Right. So that's the protection that just appeared in 2019.

Leo: Right.

Steve: And that's still not protection from this.

Leo: Okay. Well, there we have it. Steve's going to be back on Thursday, a rare appearance on a Thursday by Steve Gibson. We're doing an event with LastPass, 1:00 p.m. Pacific, 4:00 p.m. Eastern. This is much like the event we did in Boston. Big difference is we're streaming it because of COVID. So everybody's able to attend, which is great.

Steve: And Leo, I don't know, are we going to do a virtual selfie line? What are we going to do at events like this?

Leo: They'll just have to pose in front of the TV screen and take a picture.

Steve: I'm not going to be able to sign anybody's copies of SpinRite.

Leo: No, no, no. But you will be able to watch. It's going to be a panel on the future of identity, which is a really interesting topic.

Steve: Yes.

Leo: Steve Gibson; Gerry Beuchelt, who's the CISO of LogMeIn; Andrew Keen, who is really great on the political implications of all of this. It's going to be a fascinating panel. 1:00 o'clock it starts, Pacific time, 4:00 p.m. on Thursday. You can watch live at TWiT.tv/live or listen live. And it will then after the fact be put up on the TWiT events feed. And we're going to - I should tell people we're going to give you a chance to ask questions using Twitter. So we'll have a hashtag. I don't think we know what it is yet. But we'll have a hashtag which you can then ask questions on Twitter as you listen to the presentation, and we'll address those in the last half hour of the panel. So see you Thursday, 1:00 p.m. Pacific, 4:00 p.m. Eastern time. Thank you, Steve.

Steve: Thanks, buddy.

Leo: If you want to get Security Now!, the easiest thing to do is to go to Steve's site, GRC.com, 16Kb versions, 64Kb versions, even carefully crafted transcriptions of the show exist there, along with SpinRite and ShieldsUP! and all sorts of great stuff. GRC.com. He's on Twitter, @SGgrc. You can DM him there. He allows DMs from everybody. So that's a good place to ask questions. Or GRC.com/feedback.

We have copies of the show at our website, TWiT.tv/sn. Best thing to do, subscribe. That way you'll get it the minute it's available, every Tuesday. We do the show 1:30 p.m. Pacific, 4:30 Eastern, 20:30 UTC on Tuesdays at TWiT.tv/live.

Thanks, Steve. And stay safe, stay healthy. We'll be back here Thursday and then again next Tuesday for Security Now!.

Steve: Perfect. Bye.